PL/SQL TASKS

DONE BY,
P.DILIPKUMAR

Task 1 (Create Table, Insert Data to Table, Write Queries)

Create the below table.

Policy Number	Varchar2(20)
Plan Code	Varchar2(10)
Age	Number
Sum Assured	Number
Premium	Number
Start Date	Date
End Date	Date

Insert below data to the table.

Policy Number	Plan Code	Age	Sum Assured	Premium	Start Date	End Date
POL_1	P1	30	50000	500	01/01/2020	31/12/2020
POL_2	P1	32	80000	800	05-Jan-2020	04-Jan-2021
POL_3	P2	40	100000	100	01-02-2020	31-01-2020
POL_4	P1	38	250000	2500	15-02-2020	14-02-2020
PL_5	P2	41	700000	700	21st Jan 2020	20th Jan 2021

Create a table Insurance

```
DROP TABLE INSURANCE;

CREATE TABLE INSURANCE (
POLICY_NUMBER VARCHAR2(10),
PLAN_CODE VARCHAR2(10),
AGE NUMBER,
SUM_ASSURED NUMBER,
PREMIUM NUMBER,
START_DATE DATE,
END_DATE DATE
);
```

Insert record into a insurance table

```
INSERT INTO
INSURANCE(POLICY_NUMBER,PLAN_CODE,AGE,SUM_ASSURED,PREMIUM,START_DATE,END_DATE)
VALUES('POL_1','P1' ,30,50000, 500, '01-JAN-2020', '31-DEC-2020');
```

```
INSERT INTO
INSURANCE(POLICY_NUMBER,PLAN_CODE,AGE,SUM_ASSURED,PREMIUM,START_DATE,END_DATE)
VALUES('POL_2','P1' ,32,80000  ,800 ,'05-JAN-2020', '04-JAN-2021');

INSERT INTO
INSURANCE(POLICY_NUMBER,PLAN_CODE,AGE,SUM_ASSURED,PREMIUM,START_DATE,END_DATE)
VALUES('POL_3','P2' ,40,100000 ,100 ,'01-FEB-2020', '31-JAN-2020');

INSERT INTO
INSURANCE(POLICY_NUMBER,PLAN_CODE,AGE,SUM_ASSURED,PREMIUM,START_DATE,END_DATE)
VALUES('POL_4','P1',38 ,250000 ,2500 ,'15-FEB-2020' ,'14-FEB-2021');

INSERT INTO
INSURANCE(POLICY_NUMBER,PLAN_CODE,AGE,SUM_ASSURED,PREMIUM,START_DATE,END_DATE)
VALUES('PL_5',' P2', 41, 700000, 700 ,'21-JAN- 2020', '20-JAN-2021');

SELECT * FROM INSURANCE;

ALTER TABLE INSURANCE ADD CONSTRAINTS INSURANCE_PK PRIMARY KEY(POLICY_NUMBER);

DESC INSURANCE;
```

Write Queries for the below:-

1. Plan Code Wise Policy Count
2. Plan Code Wise Total Premium, Total Sum Assured
3. Find policies with age between 30 and 40
4. Find Policies with policy number starting with PL
5. Select Policies age wise in descending order
6. Plan Code wise count of policies where age greater than 30
7. Find policies with start date in January
8. Select policies premium wise ascending, sum assured wise descending
9. Plan Code wise Total Premium greater than 1000
10. Plan Code Wise Total Premium less than 3000 and Total Sum Assured greater than 50000

/*QUERIES*/

1. Plan Code Wise Policy Count

SQL Code:

```
SELECT  PLAN_CODE,COUNT(POLICY_NUMBER) FROM INSURANCE
GROUP BY PLAN_CODE
ORDER BY PLAN_CODE;
```

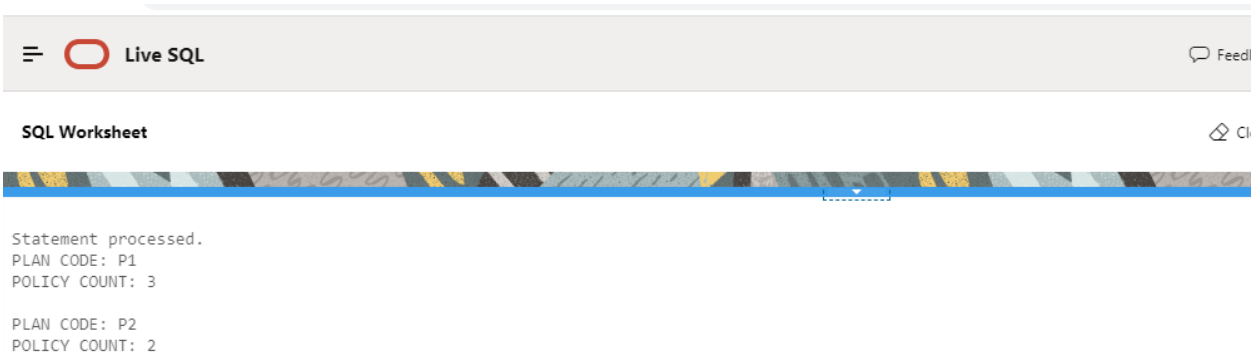
Plan Code Wise Policy Count using PL/SQL Cursors

```
DECLARE
   v_plancode INSURANCE.PLAN_CODE%TYPE;
          v_count NUMBER;
          CURSOR v_insur_cur IS SELECT PLAN_CODE,COUNT(*)
   FROM INSURANCE
   GROUP BY PLAN_CODE
  ORDER BY PLAN_CODE;
BEGIN
   OPEN v_insur_cur;
LOOP
   FETCH v_insur_cur INTO v_plancode, v_count;
   EXIT WHEN v_insur_cur%NOTFOUND;
          DBMS_OUTPUT.PUT_LINE('PLAN CODE: '||v_plancode);
          DBMS_OUTPUT.PUT_LINE('POLICY COUNT: '||v_count);
          DBMS_OUTPUT.PUT_LINE('  ');
END LOOP;
          CLOSE v_insur_cur;
END;
```

Plan Code Wise Policy Count using PL/SQL Cursor For loop

```
DECLARE
          CURSOR v_insur_cur IS SELECT PLAN_CODE,COUNT(*) AS "POLICY_COUNT"
   FROM INSURANCE
   GROUP BY PLAN_CODE
          ORDER BY PLAN_CODE;
BEGIN
   FOR insurance_rec IN v_insur_cur
   LOOP
   EXIT WHEN v_insur_cur%NOTFOUND;
          DBMS_OUTPUT.PUT_LINE('PLAN CODE: '||insurance_rec.PLAN_CODE);
          DBMS_OUTPUT.PUT_LINE('POLICY COUNT: '||insurance_rec.POLICY_COUNT);
          DBMS_OUTPUT.PUT_LINE('  ');
   END LOOP;
END;
```

Result:


Live SQL
Feedl
SQL Worksheet
Cl
Statement processed.
PLAN CODE: P1
POLICY COUNT: 3
PLAN CODE: P2
POLICY COUNT: 2

2. Plan Code Wise Total Premium, Total Sum Assured

SQL Code:

```
SELECT PLAN_CODE,SUM(PREMIUM),SUM(SUM_ASSURED)
FROM INSURANCE
GROUP BY PLAN_CODE
ORDER BY PLAN_CODE;
```

Plan Code Wise Total Premium, Total Sum Assured using PL/SQL Cursor

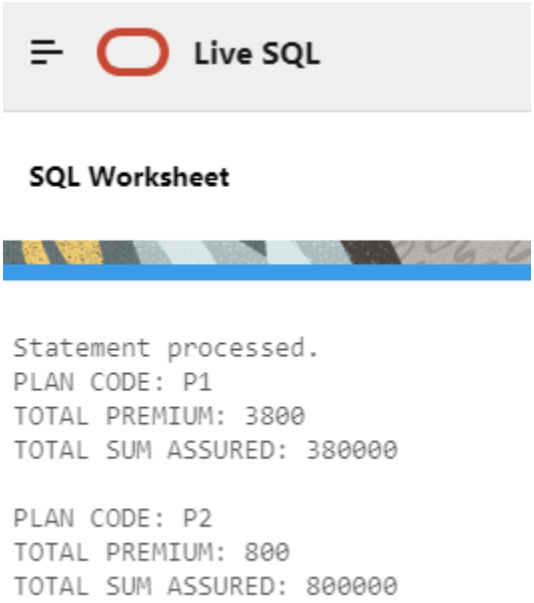
```
DECLARE
   v_plancode INSURANCE.PLAN_CODE%TYPE;
   v_totalsumassured NUMBER;
   v_totalpremium NUMBER;
         CURSOR v_insur_cur IS SELECT  PLAN_CODE,SUM(PREMIUM),SUM(SUM_ASSURED)
         FROM INSURANCE
         GROUP BY PLAN_CODE
   ORDER BY PLAN_CODE;
BEGIN
   OPEN v_insur_cur;
LOOP
   FETCH v_insur_cur INTO v_plancode,v_totalpremium,v_totalsumassured;
   EXIT WHEN v_insur_cur%NOTFOUND;
         DBMS_OUTPUT.PUT_LINE('PLAN CODE: '||v_plancode);
         DBMS_OUTPUT.PUT_LINE('TOTAL PREMIUM : '||v_totalpremium);
         DBMS_OUTPUT.PUT_LINE('TOTAL SUM ASSURED : '||v_totalsumassured);
         DBMS_OUTPUT.PUT_LINE('   ');
END LOOP;
         CLOSE v_insur_cur;
END;
```

Plan Code Wise Total Premium, Total Sum Assured using PL/SQL Cursor For loop

```
DECLARE
   CURSOR v_insur_cur IS SELECT  PLAN_CODE,SUM(PREMIUM) AS
"TOTAL_PREMIUM",SUM(SUM_ASSURED) AS "TOTAL_ASSUERD"
         FROM INSURANCE
         GROUP BY PLAN_CODE
   ORDER BY PLAN_CODE;
BEGIN
   FOR insur_rec IN v_insur_cur
         LOOP
   EXIT WHEN v_insur_cur%NOTFOUND;
         DBMS_OUTPUT.PUT_LINE('PLAN CODE: '||insur_rec.PLAN_CODE);
         DBMS_OUTPUT.PUT_LINE('TOTAL PREMIUM: '||insur_rec.TOTAL_PREMIUM);
         DBMS_OUTPUT.PUT_LINE('TOTAL SUM ASSURED: '||insur_rec.TOTAL_ASSUERD);
         DBMS_OUTPUT.PUT_LINE('  ');
   END LOOP;
```

END;

Result:



3. Find policies with age between 30 and 40

SQL code:

```
SELECT POLICY_NUMBER, AGE
FROM INSURANCE
WHERE AGE BETWEEN 30 AND 40;
```

Find policies with age between 30 and 40 using PL/SQL Cursor

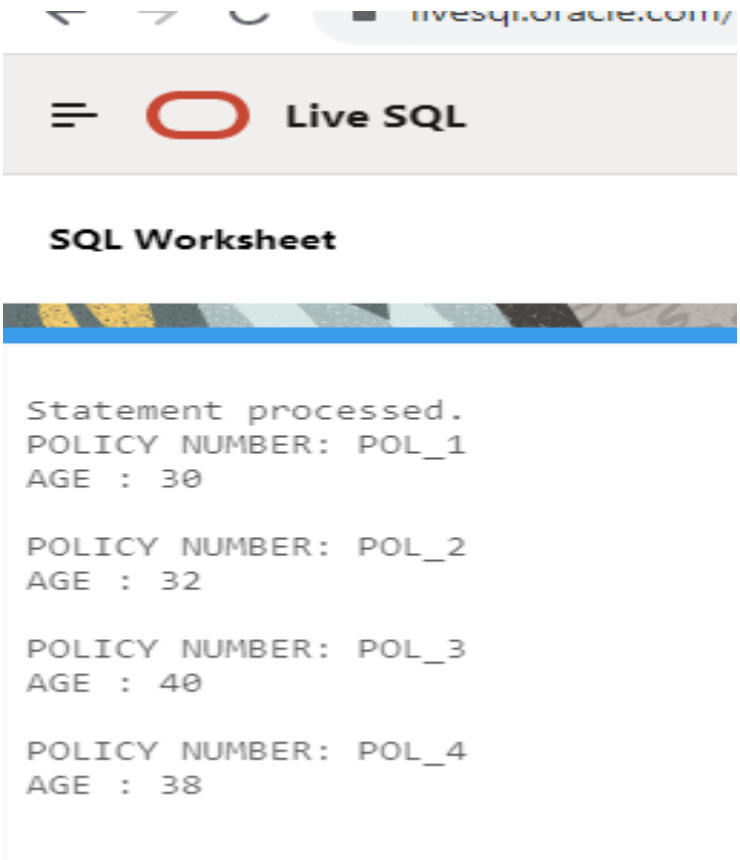
```
DECLARE
   v_policynumber INSURANCE.POLICY_NUMBER%TYPE;
          v_age INSURANCE.AGE%TYPE;
          CURSOR v_insur_cur IS SELECT POLICY_NUMBER, AGE
   FROM INSURANCE
   WHERE AGE BETWEEN 30 AND 40;
BEGIN
   OPEN v_insur_cur ;
          LOOP
   FETCH v_insur_cur INTO v_policynumber,v_age;
    EXIT WHEN v_insur_cur%NOTFOUND;
          DBMS_OUTPUT.PUT_LINE('POLICY NUMBER: '||v_policynumber);
          DBMS_OUTPUT.PUT_LINE('AGE : '||v_age);
          DBMS_OUTPUT.PUT_LINE('  ');
          END LOOP;
```

```
        CLOSE v_insur_cur ;
END;
```

Find policies with age between 30 and 40 using PL/SQL Cursor For loop

```
DECLARE
  CURSOR v_insur_cur IS SELECT POLICY_NUMBER, AGE
   FROM INSURANCE
   WHERE AGE BETWEEN 30 AND 40;
BEGIN
   FOR insur_rec IN v_insur_cur
        LOOP
   EXIT WHEN v_insur_cur%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('POLICY NUMBER: '||insur_rec.POLICY_NUMBER);
        DBMS_OUTPUT.PUT_LINE('AGE : '||insur_rec.AGE);
        DBMS_OUTPUT.PUT_LINE('  ');
   END LOOP;
END;
```

Result:



4. Find Policies with policy number starting with PL

SQL Code:

```
SELECT POLICY_NUMBER FROM INSURANCE
WHERE POLICY_NUMBER LIKE'PL%';
```

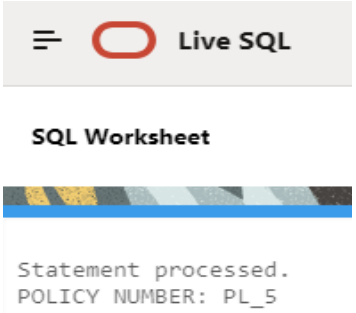
Find Policies with policy number starting with PL using PL/SQL Cursor

```
DECLARE
   v_policynumber INSURANCE.POLICY_NUMBER%TYPE;
          CURSOR v_insur_cur IS SELECT POLICY_NUMBER FROM INSURANCE
          WHERE POLICY_NUMBER LIKE'PL%';
BEGIN
   OPEN v_insur_cur ;
LOOP
   FETCH v_insur_cur INTO v_policynumber;
          EXIT WHEN v_insur_cur%NOTFOUND;
          DBMS_OUTPUT.PUT_LINE('POLICY NUMBER: '||v_policynumber);
   END LOOP;
CLOSE v_insur_cur ;
END;
```

Find Policies with policy number starting with PL using PL/SQL Cursor For loop

```
DECLARE
   CURSOR v_insur_cur IS SELECT POLICY_NUMBER
   FROM INSURANCE
   WHERE POLICY_NUMBER LIKE '%PL%';
BEGIN
  FOR insur_rec IN v_insur_cur
   LOOP
   EXIT WHEN v_insur_cur%NOTFOUND;
          DBMS_OUTPUT.PUT_LINE('POLICY NUMBER: '||insur_rec.POLICY_NUMBER);
   END LOOP;
END;
```

Result:



5. **Select Policies age wise in descending order**

SQL code:

```
SELECT POLICY_NUMBER,AGE
FROM INSURANCE
ORDER BY AGE DESC;
```

Select Policies age wise in descending order using PL/SQL Cursor

```
DECLARE
   v_policynumber INSURANCE.POLICY_NUMBER%TYPE;
   v_age INSURANCE.AGE%TYPE;
          CURSOR v_insur_cur  IS SELECT POLICY_NUMBER,AGE
          FROM INSURANCE
          ORDER BY AGE DESC;
BEGIN
   OPEN v_insur_cur;
          LOOP
   FETCH v_insur_cur INTO v_policynumber,v_age;
          EXIT WHEN v_insur_cur%NOTFOUND;
          DBMS_OUTPUT.PUT_LINE('POLICY NUMBER: '||v_policynumber);
          DBMS_OUTPUT.PUT_LINE('AGE : '||v_age);
          DBMS_OUTPUT.PUT_LINE('    ');
          END LOOP;
          CLOSE v_insur_cur;
END;
```

Select Policies age wise in descending order using PL/SQL Cursor For loop

```
DECLARE
CURSOR v_insur_cur  IS SELECT POLICY_NUMBER,AGE
          FROM INSURANCE
          ORDER BY AGE DESC;
BEGIN
          FOR ins_rec IN v_insur_cur
   LOOP
      EXIT WHEN v_insur_cur%NOTFOUND;
          DBMS_OUTPUT.PUT_LINE('POLICY NUMBER: '||ins_rec.POLICY_NUMBER);
          DBMS_OUTPUT.PUT_LINE('AGE : '||ins_rec.AGE);
          DBMS_OUTPUT.PUT_LINE('    ');
   END LOOP;
END;
```

Result:

Live SQL

SQL Worksheet

```
Statement processed.
POLICY NUMBER: PL_5
AGE : 41

POLICY NUMBER: POL_3
AGE : 40

POLICY NUMBER: POL_4
AGE : 38

POLICY NUMBER: POL_2
AGE : 32

POLICY NUMBER: POL_1
AGE : 30
```

6. Plan Code wise count of policies where age greater than 30

SQL Code:

```
SELECT PLAN_CODE, COUNT (POLICY_NUMBER) AS POLICY_COUNT FROM INSURANCE
WHERE AGE>30
GROUP BY PLAN_CODE
ORDER BY PLAN_CODE;
```

Plan Code wise count of policies where age greater than 30 using PL/SQL Cursor

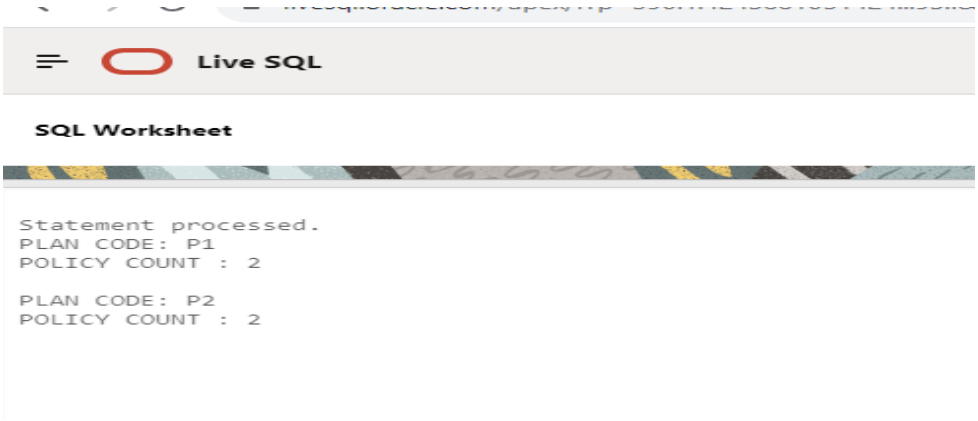
```
DECLARE
   v_plancode INSURANCE.PLAN_CODE%TYPE;
   v_count NUMBER;
         CURSOR v_insur_cur IS  SELECT PLAN_CODE, COUNT (POLICY_NUMBER)
```

```
   FROM INSURANCE
   WHERE AGE>30
   GROUP BY PLAN_CODE
   ORDER BY PLAN_CODE;
BEGIN
   OPEN v_insur_cur;
          LOOP
   FETCH v_insur_cur INTO v_plancode,v_count;
          EXIT WHEN v_insur_cur%NOTFOUND;
          DBMS_OUTPUT.PUT_LINE('PLAN CODE : '||v_plancode);
          DBMS_OUTPUT.PUT_LINE(COUNT : '||v_count);
          DBMS_OUTPUT.PUT_LINE('    ');
          END LOOP;
          CLOSE v_insur_cur;
END;
```

Plan Code wise count of policies where age greater than 30 using PL/SQL Cursor For loop

```
DECLARE
CURSOR v_insur_cur IS SELECT PLAN_CODE, COUNT (POLICY_NUMBER) AS "POLICY_COUNT"
FROM INSURANCE
WHERE AGE>30
GROUP BY PLAN_CODE
ORDER BY PLAN_CODE;
BEGIN
   FOR ins_rec IN v_insur_cur
LOOP
 EXIT WHEN v_insur_cur%NOTFOUND;
          DBMS_OUTPUT.PUT_LINE('PLAN CODE: '||ins_rec.PLAN_CODE);
          DBMS_OUTPUT.PUT_LINE('POLICY COUNT : '||ins_rec.POLICY_COUNT);
          DBMS_OUTPUT.PUT_LINE('    ');
   END LOOP;
END;
```

Result:



7. Find policies with start date in January

SQL code:

```
SELECT POLICY_NUMBER, START_DATE FROM INSURANCE
WHERE START_DATE LIKE '%JAN%';
```

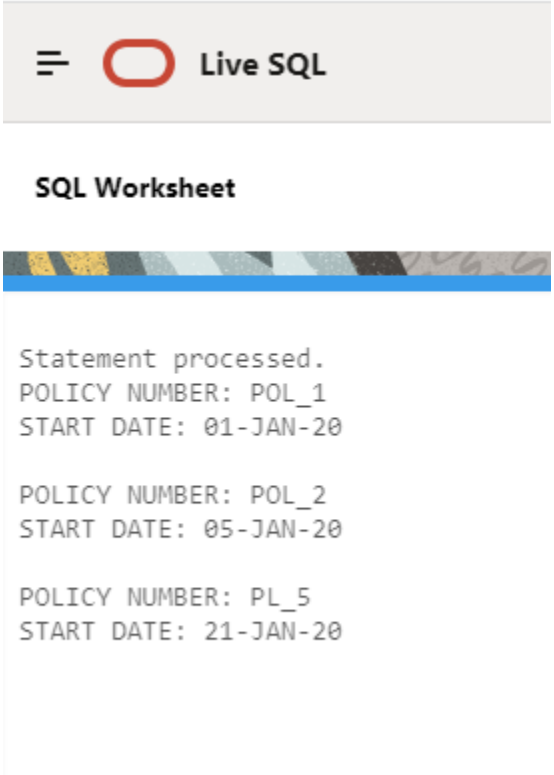
Find policies with start date in January using PL/SQL cursor

```
DECLARE
   v_policynumber INSURANCE.POLICY_NUMBER%TYPE;
          v_startdate INSURANCE.START_DATE%TYPE;
          CURSOR v_insur_cur IS SELECT POLICY_NUMBER, START_DATE FROM INSURANCE
          WHERE START_DATE LIKE '%JAN%';
BEGIN
   OPEN v_insur_cur;
LOOP
   FETCH v_insur_cur INTO v_policynumber,v_startdate;
EXIT WHEN v_insur_cur%NOTFOUND;
DBMS_OUTPUT.PUT_LINE('POLICY NUMBER: '||v_policynumber);
DBMS_OUTPUT.PUT_LINE('START DATE: '||v_startdate);
DBMS_OUTPUT.PUT_LINE('    ');
END LOOP;
CLOSE v_insur_cur;
END;
```

Find policies with start date in January using PL/SQL cursor

```
DECLARE
CURSOR v_insur_cur IS SELECT POLICY_NUMBER, START_DATE FROM INSURANCE
          WHERE START_DATE LIKE '%JAN%';
BEGIN
  FOR ins_rec IN v_insur_cur
LOOP
EXIT WHEN v_insur_cur%NOTFOUND;
DBMS_OUTPUT.PUT_LINE('POLICY NUMBER: '||ins_rec.POLICY_NUMBER);
DBMS_OUTPUT.PUT_LINE('START DATE: '||ins_rec.START_DATE);
DBMS_OUTPUT.PUT_LINE('    ');
END LOOP;
END;
```

Result:



8. Select policies premium wise ascending, sum assured wise descending

SQL code:

```
SELECT POLICY_NUMBER, PREMIUM, SUM_ASSURED FROM INSURANCE
ORDER BY PREMIUM, SUM_ASSURED DESC;
```

Select policies premium wise ascending, sum assured wise descending using PL/SQL cursor

```
DECLARE
v_policynumber INSURANCE.POLICY_NUMBER%TYPE;
v_premium INSURANCE.PREMIUM        %TYPE;
v_summassured INSURANCE.SUM_ASSURED%TYPE;
CURSOR v_ins_cur IS SELECT POLICY_NUMBER,PREMIUM,SUM_ASSURED
FROM INSURANCE
ORDER BY PREMIUM ASC,
SUM_ASSURED DESC;
BEGIN
   OPEN v_ins_cur;
LOOP
   FETCH v_ins_cur INTO v_policynumber,v_premium,v_summassured ;
          EXIT WHEN v_ins_cur%NOTFOUND;
          DBMS_OUTPUT.PUT_LINE('POLICY NUMBER : '||v_policynumber);
```

```
        DBMS_OUTPUT.PUT_LINE('PREMIUM : '||v_premium);
        DBMS_OUTPUT.PUT_LINE('SUM ASSURED: '||v_summassured);
        DBMS_OUTPUT.PUT_LINE('    ');
  END LOOP;
CLOSE v_ins_cur;
END;
```

Select policies premium wise ascending, sum assured wise descending using PL/SQL cursor for loop

```
DECLARE
CURSOR v_ins_cur IS SELECT POLICY_NUMBER,PREMIUM,SUM_ASSURED
FROM INSURANCE
ORDER BY PREMIUM ASC,
SUM_ASSURED DESC;
BEGIN
  FOR ins_rec IN v_ins_cur
LOOP
        EXIT WHEN v_ins_cur%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('POLICY NUMBER : '||ins_rec.POLICY_NUMBER);
        DBMS_OUTPUT.PUT_LINE('PREMIUM : '||ins_rec.PREMIUM);
        DBMS_OUTPUT.PUT_LINE('SUM ASSURED: '||ins_rec.SUM_ASSURED);
        DBMS_OUTPUT.PUT_LINE('    ');
  END LOOP;
END;
```

Result:

9. Plan Code wise Total Premium greater than 1000

SQL code:

```
SELECT PLAN_CODE, SUM (PREMIUM) AS TOTAL_PREMIUM FROM INSURANCE
GROUP BY PLAN_CODE
HAVING SUM (PREMIUM)>1000
ORDER BY PLAN_CODE;
```

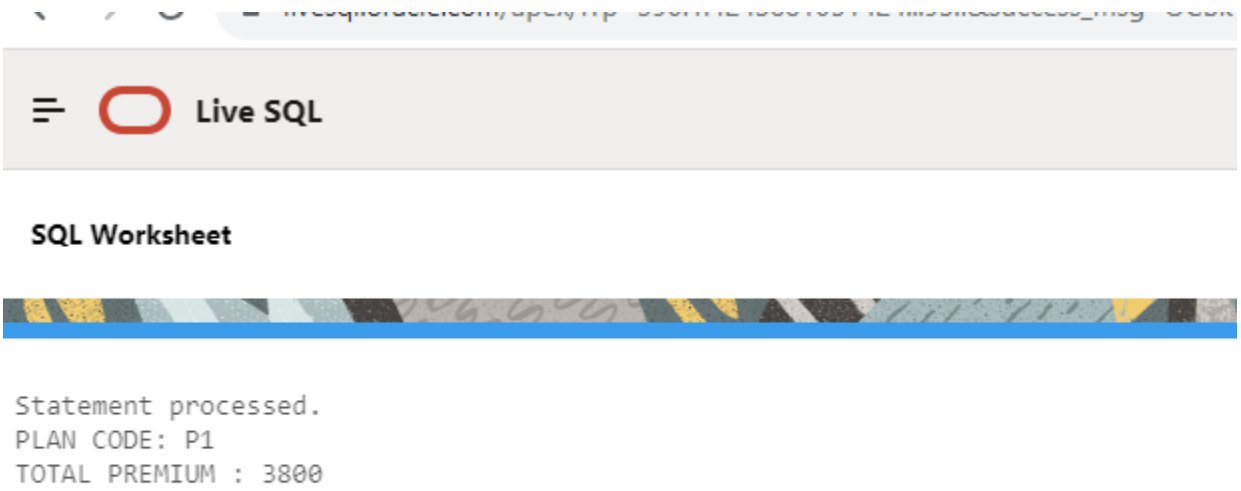
Plan Code wise Total Premium greater than 1000 using PL/SQL cursor

```
DECLARE
   v_plancode  INSURANCE.PLAN_CODE%TYPE;
   v_totalprem NUMBER;
CURSOR v_ins_cur IS SELECT PLAN_CODE, SUM (PREMIUM)
FROM INSURANCE
GROUP BY PLAN_CODE
HAVING SUM (PREMIUM)>1000
ORDER BY PLAN_CODE;
BEGIN
   OPEN v_ins_cur;
LOOP
   FETCH v_ins_cur INTO v_plancode,v_totalprem;
EXIT WHEN v_ins_cur%NOTFOUND;
DBMS_OUTPUT.PUT_LINE('PLAN CODE: '||v_plancode);
DBMS_OUTPUT.PUT_LINE('TOTAL PREMIUM : '||v_totalprem);
DBMS_OUTPUT.PUT_LINE('   ');
END LOOP;
CLOSE v_ins_cur;
END;
```

Plan Code wise Total Premium greater than 1000 using PL/SQL cursor for loop

```
DECLARE
CURSOR v_ins_cur IS SELECT PLAN_CODE, SUM (PREMIUM) AS "TOTAL_PREMIUM"
FROM INSURANCE
GROUP BY PLAN_CODE
HAVING SUM (PREMIUM)>1000
ORDER BY PLAN_CODE;
BEGIN
 FOR ins_rec IN v_ins_cur
LOOP
EXIT WHEN v_ins_cur%NOTFOUND;
DBMS_OUTPUT.PUT_LINE('PLAN CODE: '||ins_rec.PLAN_CODE);
DBMS_OUTPUT.PUT_LINE('TOTAL PREMIUM : '||ins_rec.TOTAL_PREMIUM);
DBMS_OUTPUT.PUT_LINE('   ');
END LOOP;
END;
```

Result:



10. Plan Code Wise Total Premium less than 3000 and Total Sum Assured greater than 50000

SQL code:

```
SELECT PLAN_CODE,SUM(PREMIUM),SUM(SUM_ASSURED)
FROM INSURANCE
GROUP BY PLAN_CODE
HAVING SUM(PREMIUM)<3000 AND SUM(SUM_ASSURED)>50000
ORDER BY PLAN_CODE;
```

Plan Code Wise Total Premium less than 3000 and Total Sum Assured greater than 50000 using PL/SQL cursor

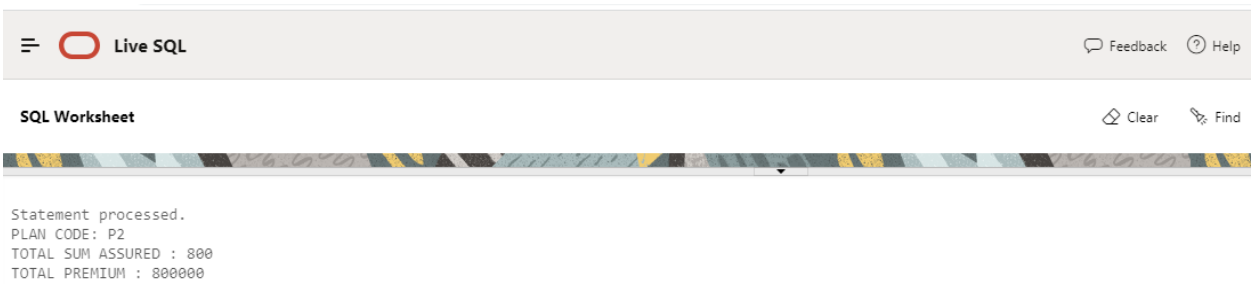
```
DECLARE
  v_plancode  INSURANCE.PLAN_CODE%TYPE;
        v_totalsumassured NUMBER;
  v_totalprem NUMBER;
CURSOR v_ins_cur IS SELECT PLAN_CODE,SUM(PREMIUM),SUM(SUM_ASSURED)
  FROM INSURANCE
  GROUP BY PLAN_CODE
```

```
   HAVING SUM(PREMIUM)<3000 AND SUM(SUM_ASSURED)>50000
   ORDER BY PLAN_CODE;
BEGIN
   OPEN  v_ins_cur ;
LOOP
   FETCH  v_ins_cur INTO v_plancode,v_totalsumassured,v_totalprem;
   EXIT WHEN  v_ins_cur%NOTFOUND;
DBMS_OUTPUT.PUT_LINE('PLAN CODE: '||v_plancode);
DBMS_OUTPUT.PUT_LINE('TOTAL SUM ASSURED : '||v_totalsumassured);
DBMS_OUTPUT.PUT_LINE('TOTAL PREMIUM : '||v_totalprem);
   END LOOP;
CLOSE  v_ins_cur ;
END;
```

Plan Code Wise Total Premium less than 3000 and Total Sum Assured greater than 50000 using PL/SQL cursor for loop

```
DECLARE
CURSOR v_ins_cur IS SELECT PLAN_CODE,SUM(PREMIUM) AS "TOT_PREM",SUM(SUM_ASSURED) AS
"TOT_SUM_ASSURED"
   FROM INSURANCE
   GROUP BY PLAN_CODE
   HAVING SUM(PREMIUM)<3000 AND SUM(SUM_ASSURED)>50000
   ORDER BY PLAN_CODE;
BEGIN
FOR ins_rec IN v_ins_cur
LOOP
   EXIT WHEN  v_ins_cur%NOTFOUND;
DBMS_OUTPUT.PUT_LINE('PLAN CODE: '||ins_rec.PLAN_CODE);
DBMS_OUTPUT.PUT_LINE('TOTAL SUM ASSURED : '||ins_rec.TOT_SUM_ASSURED);
DBMS_OUTPUT.PUT_LINE('TOTAL PREMIUM : '||ins_rec.TOT_PREM);
   END LOOP;
END;
```

Result:



Task 2 (Create Tables, Insert Data to Table 1, Write Procedure to Populate Table 2)

Create 2 tables Class Grade and Student Grade
Table - Class Grade

Class	Mark From	Mark To	Grade
1	1	39	Fail
1	40	49	3rd Class
1	50	59	2nd Class
1	60	79	1st Class
1	80	100	Distinction

Insert above date into Class Grade table.

Table – Student Grade

Student	Class	Mark	Grade

Write a procedure with input (Student, Class and Mark)
Find the Grade from Class Grade table based on
Class & Mark

Insert Input data and grade into Student Grade Table

PL/SQL code:

Create a table class_grade

```
DROP TABLE CLASS_GRADE;
CREATE TABLE CLASS_GRADE
(
CLASS NUMBER,
MARK_FROM NUMBER,
MARK_TO NUMBER,
GRADE VARCHAR2(50)
);
```

Insert values into the class_grade table

```
INSERT INTO CLASS_GRADE(CLASS,MARK_FROM,MARK_TO,GRADE) VALUES (1,1,39,'FAIL');
INSERT INTO CLASS_GRADE(CLASS,MARK_FROM,MARK_TO,GRADE) VALUES (1,40,49,'3RD CLASS');
INSERT INTO CLASS_GRADE(CLASS,MARK_FROM,MARK_TO,GRADE) VALUES (1,50,59,'2ND CLASS');
```

```
INSERT INTO CLASS_GRADE(CLASS,MARK_FROM,MARK_TO,GRADE) VALUES (1,60,79,'1ST  CLASS');
INSERT INTO CLASS_GRADE(CLASS,MARK_FROM,MARK_TO,GRADE) VALUES (1,80,100,'DISTINCTION');

SELECT * FROM CLASS_GRADE;
DROP TABLE STUDENT_GRADE;
```

Create a table student_grade

```
CREATE TABLE STUDENT_GRADE
(
STUDENT NUMBER,
CLASS NUMBER,
MARK NUMBER,
GRADE VARCHAR2(50)
);

SELECT * FROM STUDENT_GRADE;
```

Create a procedure for finding grade result

```
CREATE OR REPLACE PROCEDURE student_mark_procedure(v_student NUMBER,
   v_class NUMBER,
   v_mark NUMBER
   )
AS
v_graderesult VARCHAR2(50);
BEGIN
SELECT GRADE INTO v_graderesult
FROM CLASS_GRADE
WHERE CLASS = v_class
AND MARK_FROM <=  v_mark
AND MARK_TO  >= v_mark;
INSERT INTO STUDENT_GRADE (STUDENT,CLASS, MARK,GRADE) VALUES
(v_student,v_class,v_mark,v_graderesult) ;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
   DBMS_OUTPUT.PUT_LINE('No data found, please check to the class grade table');
          WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('An Error occured...'||SQLERRM);
END student_mark_procedure;
```

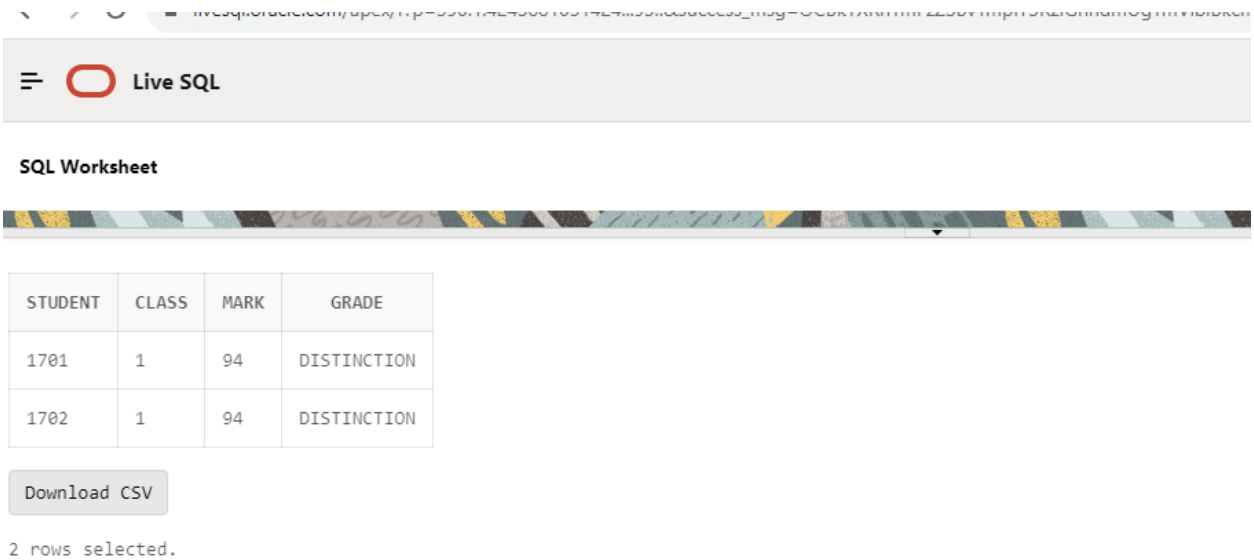
Execute a procedure

```
BEGIN
   student_mark_procedure(1701,1,94);
END;
```

EXEC student_mark_procedure(1702,1,94);

EXEC student_mark_procedure(1703,1,-5);

Result:



STUDENT	CLASS	MARK	GRADE
1701	1	94	DISTINCTION
1702	1	94	DISTINCTION

Task 3 (Create Tables, Insert Data to Table 1, Write Procedure to Populate Table 2)

Create 2 tables inventory master and inventory purchase

Table - Tbl_inventory_master

Item Code	Item Name	Stock	Selling Rate	Buying Rate
IT01	Pen	0	10	6
IT02	Pencil	0	5	3
IT03	Eraser	0	3	1
IT04	Scale	0	25	15

Create DML Statements to insert the above data to inventory master table.

Table - Tbl_inventory_purchase

Item code	Purchase date	Purchase Qty	Selling Rate	Purchase Amt

Write a Procedure to insert data into inventory purchase table by passing Item Code, Purchase Date and Purchase quantity as parameters.

Before inserting into above table find the selling rate of that item code from tbl_inventory_master and calculate purchase amount (rate * purchase qty)

While Inserting into the above table, insert input data along with selling rate and calculated purchase amount

Create a table Inventory_master

```
DROP TABLE INVENTORY_MASTER;

CREATE TABLE INVENTORY_MASTER
(
ITEM_CODE VARCHAR2(50),
ITEM_NAME VARCHAR2(200),
STOCK NUMBER,
SELLING_RATE NUMBER,
BUYING_RATE NUMBER
```

```
);

SELECT * FROM INVENTORY_MASTER;
```

Insert values into the table inventory_master

```
INSERT INTO INVENTORY_MASTER(ITEM_CODE,ITEM_NAME ,STOCK ,SELLING_RATE ,BUYING_RATE )
VALUES ( 'IT01','Pen',0,10,6);
INSERT INTO INVENTORY_MASTER(ITEM_CODE,ITEM_NAME ,STOCK ,SELLING_RATE ,BUYING_RATE )
VALUES ( 'IT02','PENCIL',0,5,3);
INSERT INTO INVENTORY_MASTER(ITEM_CODE,ITEM_NAME ,STOCK ,SELLING_RATE ,BUYING_RATE )
VALUES ( 'IT04','ERASER',0,3,1);
INSERT INTO INVENTORY_MASTER(ITEM_CODE,ITEM_NAME ,STOCK ,SELLING_RATE ,BUYING_RATE )
VALUES ( 'IT04','SCLAE',0,25,15);
```

Create a table Inventory_purchase

```
DROP TABLE INVENTORY_PURCHASE;
CREATE TABLE INVENTORY_PURCHASE
(
ITEM_CODE VARCHAR2(50),
PURCHASE_DATE DATE,
PURCHASE_QTY NUMBER,
SELLING_RATE NUMBER,
PURCHASE_AMT  NUMBER
);
```

Create a procedure to insert inventery_purchase table

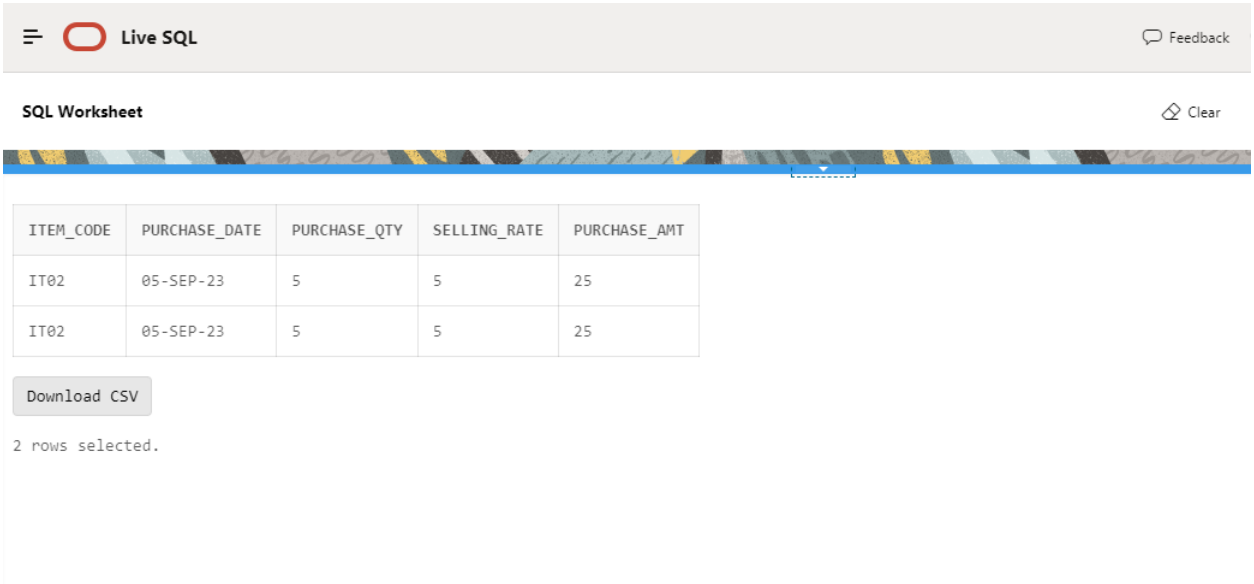
```
DROP PROCEDURE insert_inventory;

CREATE OR REPLACE PROCEDURE insert_inventory
(
   v_itemcode  INVENTORY_MASTER.ITEM_CODE%TYPE,
   v_purchase_date DATE,
   v_purchase_qty NUMBER
)
AS
v_selling_rate NUMBER;
v_purchase_amt NUMBER;
BEGIN
   SELECT SELLING_RATE
   INTO v_selling_rate
   FROM INVENTORY_MASTER
   WHERE ITEM_CODE = v_itemcode;
   v_purchase_amt := v_selling_rate * v_purchase_qty;
```

```
INSERT INTO INVENTORY_PURCHASE
(ITEM_CODE,PURCHASE_DATE,PURCHASE_QTY,SELLING_RATE,PURCHASE_AMT) VALUES
(v_itemcode,v_purchase_date,v_purchase_qty,v_selling_rate,v_purchase_amt);
END insert_inventory;
```

Execute a procedure:

```
EXECUTE insert_inventory('IT02',SYSDATE,5);
```

Result:



ITEM_CODE	PURCHASE_DATE	PURCHASE_QTY	SELLING_RATE	PURCHASE_AMT
IT02	05-SEP-23	5	5	25
IT02	05-SEP-23	5	5	25

Task 4 (Create Tables, Write Procedure to Populate both the Tables)

Create 2 tables card master and card details

Card Master

Card ID	Varchar2(20)
Card Name	Varchar2(100)
Prev Type	Varchar2(1)
Prev Amount	Number
Curr Type	Varchar2(1)
Curr Amount	Number
Net Type	Varchar2(1)
Net Amount	Number
Last_trans_Date	Date

Card Details

Card ID	Varchar2(20)
Type	Varchar2(1)
Amount	Number
Trans_Date	Date

Write a procedure with the following parameters
Card ID, Card Name, Type, Amount, Trans Date

If it is new card ID, a new record should be created in Card Master table.
Prev Type, Prev Amount should be Null.
Current Type, Current Amount should be Type and Amount from Input.
Net Type and Net amount should be Type and Amount from Input.
Last Trans Date should be Trans Date from Input.

Card ID, Type, Amount, Trans date should be inserted into Card details table as it is from Input.

Exec Procedure P1 (C1, ‘John’, D, 1000, ‘10/10/2021’);
Exec Procedure P1 (C2, ‘Raj’, D, 500, ‘12/10/2021’);
Exec Procedure P1 (C3, ‘Williams ‘, C, 100, ‘15/10/2021’);

Above execution, should give the below results in card master and card details tables.

Card ID	Card Name	Prev Type	Prev Amount	Curr Type	Curr Amount	Net Type	Net Amount	Last Trans Date
C1	John			D	1000	D	1000	10/10/2021
C2	Raj			D	500	D	500	12/10/2021
C3	Williams			C	100	C	100	15/10/2021

Card ID	Type	Amount	Trans Date
C1	D	1000	10/10/2021
C2	D	500	12/10/2021
C3	C	100	15/10/2021

If the card already exists, in card master, data should be inserted as it is in card details table.
In Master existing data should be updated as below:-
Prev Type, Prev Amount, Current Type, Current Amount and Net Type and Net amount should be calculated as the examples shown in the transactions below.
Last Trans Date should be Trans Date from Input.

Exec Procedure P1 (C1, 'John', D, 200, '15/10/2021');

Card ID	Card Name	Prev Type	Prev Amount	Curr Type	Curr Amount	Net Type	Net Amount	Last Trans Date
C1	John	D	1000	D	200	D	1200	15/10/2021

Card ID	Type	Amount	Trans Date
C1	D	1000	10/10/2021
C1	D	200	15/10/2021

Exec Procedure P1 (C1, 'John', D, 500, '20/10/2021');

Card ID	Card Name	Prev Type	Prev Amount	Curr Type	Curr Amount	Net Type	Net Amount	Last Trans Date
C1	John	D	1200	D	500	D	1700	20/10/2021

Card ID	Type	Amount	Trans Date
C1	D	1000	10/10/2021
C1	D	200	15/10/2021
C1	D	500	20/10/2021

Exec Procedure P1 (C2, ‘Raj’, D, 500, ‘12/10/2021’);

Card ID	Card Name	Prev Type	Prev Amount	Curr Type	Curr Amount	Net Type	Net Amount	Last Trans Date
C2	Raj	D	500	D	500	D	1000	18/10/2021

Card ID	Type	Amount	Trans Date
C2	D	500	12/10/2021
C2	D	500	18/10/2021

Exec Procedure P1 (C2, ‘Raj’, C, 1000, ‘20/10/2021’);

Card ID	Card Name	Prev Type	Prev Amount	Curr Type	Curr Amount	Net Type	Net Amount	Last Trans Date
C2	Raj	D	1000	C	1000	N	0	20/10/2021

Card ID	Type	Amount	Trans Date
C2	D	500	12/10/2021
C2	D	500	18/10/2021
C2	C	1000	20/10/2021

Exec Procedure P1 (C3, ‘Williams ‘, C, 800, ‘22/10/2021’);

Card ID	Card Name	Prev Type	Prev Amount	Curr Type	Curr Amount	Net Type	Net Amount	Last Trans Date
C3	Williams	C	100	C	800	C	900	22/10/2021

Card ID	Type	Amount	Trans Date
C3	C	100	15/10/2021
C3	C	800	22/10/2021

Exec Procedure P1 (C3, ‘Williams ‘, D, 400, ‘25/10/2021’);

Card ID	Card Name	Prev Type	Prev Amount	Curr Type	Curr Amount	Net Type	Net Amount	Last Trans Date
C3	Williams	C	900	D	400	C	500	25/10/2021

Card ID	Type	Amount	Trans Date
C3	C	100	15/10/2021
C3	C	800	22/10/2021
C3	D	400	25/10/2021

Create a table card_master

```
DROP TABLE CARD_MASTER;
CREATE TABLE CARD_MASTER
(
CARD_ID VARCHAR2(20),
CARD_NAME VARCHAR2(100),
PREV_TPYE VARCHAR2(1),
PREV_AMOUNT NUMBER,
CURR_TYPE VARCHAR2(1),
CURR_AMOUNT NUMBER,
NET_TYPE VARCHAR2(1),
NET_AMOUNT NUMBER,
LAST_TRANS_DATE DATE
);

SELECT * FROM CARD_MASTER;
```

Create a table card_details

```
DROP TABLE CARD_DETAILS;
CREATE TABLE CARD_DETAILS
(
CARD_ID VARCHAR2(20),
TYPE VARCHAR2(1),
AMOUNT NUMBER,
TRANS_DATE DATE
);

SELECT * FROM CARD_DETAILS;
```

Create a procedure for both the tables:

```
CREATE OR REPLACE PROCEDURE card_proc (
  P_CARD_ID  VARCHAR2,
  P_CARD_NAME  VARCHAR2,
  P_TYPE  VARCHAR2,
  P_AMOUNT  NUMBER,
  P_TRANS_DATE  DATE
)
IS
  V_EXISTING_CARD NUMBER;
  V_PREV_TYPE VARCHAR2(1);
  V_PREV_AMOUNT NUMBER;
  V_CURRENT_TYPE VARCHAR2(1);
  V_CURRENT_AMOUNT NUMBER;
  V_NET_TYPE VARCHAR2(1);
  V_NET_AMOUNT NUMBER;
```

```
BEGIN
                                -- To Check if the card already exists in Card Master table
  SELECT COUNT(*) INTO V_EXISTING_CARD FROM CARD_MASTER WHERE CARD_ID = P_CARD_ID;

  IF V_EXISTING_CARD = 0 THEN
                                -- Card is new, insert a new record into Card Master table
    INSERT INTO CARD_MASTER (CARD_ID, CARD_NAME, PREV_TPYE, PREV_AMOUNT, CURR_TYPE,
CURR_AMOUNT, NET_TYPE, NET_AMOUNT, LAST_TRANS_DATE)
    VALUES (P_CARD_ID, P_CARD_NAME, NULL, NULL, P_TYPE, P_AMOUNT, P_TYPE, P_AMOUNT,
P_TRANS_DATE);
  ELSE

    SELECT CURR_TYPE, CURR_AMOUNT INTO V_PREV_TYPE, V_PREV_AMOUNT
    FROM CARD_MASTER
    WHERE CARD_ID = P_CARD_ID;
    V_CURRENT_TYPE := P_TYPE;
    V_CURRENT_AMOUNT := P_AMOUNT;
    V_NET_TYPE := P_TYPE;
    V_NET_AMOUNT := V_PREV_AMOUNT + P_AMOUNT;
                                -- Update the record in Card Master table
    UPDATE CARD_MASTER
    SET PREV_TPYE = V_PREV_TYPE,
       PREV_AMOUNT = V_PREV_AMOUNT,
       CURR_TYPE = V_CURRENT_TYPE,
       CURR_AMOUNT = V_CURRENT_AMOUNT,
       NET_TYPE = V_NET_TYPE,
       NET_AMOUNT = V_NET_AMOUNT,
       LAST_TRANS_DATE = P_TRANS_DATE
    WHERE CARD_ID = P_CARD_ID;
  END IF;                       -- Insert the transaction details into Card Details table
  INSERT INTO CARD_DETAILS (CARD_ID, TYPE, AMOUNT, TRANS_DATE)
  VALUES (P_CARD_ID, P_TYPE, P_AMOUNT, P_TRANS_DATE);

END;
```

Executing a Procedure

```
EXEC card_proc('C1', 'John', 'D', 1000, TO_DATE('10/10/2021', 'DD/MM/YYYY'));
EXEC card_proc('C2', 'Raj', 'D', 500, TO_DATE('12/10/2021', 'DD/MM/YYYY'));
EXEC card_proc('C3', 'Williams', 'C', 100, TO_DATE('15/10/2021', 'DD/MM/YYYY'));
```

Result:

Live SQL

SQL Worksheet

CARD_ID	CARD_NAME	PREV_TPYE	PREV_AMOUNT	CURR_TYPE	CURR_AMOUNT	NET_TYPE	NET_AMOUNT	LAST_TRANS_DATE
C1	John	-	-	D	1000	D	1000	10-OCT-21
C2	Raj	-	-	D	500	D	500	12-OCT-21
C3	Williams	-	-	C	100	C	100	15-OCT-21

Download CSV

3 rows selected.